

BİLGİSAYARLAR ve C++ İLE PROGRAMLAMAYA GİRİŞ

BİLGİSAYAR NEDİR?: Bilgisayar, insanın yapabileceğinden milyonlarca hatta milyarlarca kat hızlı hesaplama yapan ve/ya mantıksal kararlar verilebilen bir alettir. Süper bilgisayarlar saniyede yüz milyarlarca toplama yapabilirler ki bu bir insanın basit bir hesap makinesiyle onlarca yılda yapabileceği işlem miktarıdır.

Bilgisayarlar, bilgisayar programı denen komut setlerinin kontrolü altında verileri işlerler. Bilgisayarda çalışan bilgisayar programlarına yazılım denir. Bunları geliştiren, yazan kişilere de bilgisayar programcısı denir.

Bilgisayar 6 mantıksal bölümden oluşur:

- 1- Giriş birimi
- 2- Çıkış birimi
- 3- Bellek
- 4- Aritmetik mantık birimi Merkezi işlem birimi
- 5- Kontrol birimi
- 6- İkincil bellek

İŞLETİM SİSTEMLERİNİN EVRİMİ

İlk bilgisayarlar bir anda bir iş yapabiliyorlardı. Delikli kartlarla yazılan programlar, kart okuyucudan bilgisayara okutularak sırayla (bir anda bir iş) çalıştırıldı. Bir anda bir işin çalıştığı bu ortamda bilgisayarın tüm kaynakları verimli biçimde kullanılmamış olmaktadır. MİB çalışırken giriş ve çıkış birimleri boş beklemektedir. Daha verimli kaynak kullanımı için giderek “çoklu programlama” ya izin veren işletim sistemleri geliştirilmiştir. Bu ortamlarda bir anda birden fazla iş çalışır, ancak hala kullanıcılar programlarını kartlara delerek bilgisayara girmektedir.

1960'larda "timesharing" işletim sistemleri geliştirilmiştir. Timesharing bir ortamda kullanıcılar bilgisayara, terminaller (bir ekran ve klavye) aracılığıyla erişirler. Böylelikle onlarca hatta yüzlerce kişi aynı anda tek bir bilgisayara erişebilmekte ve kullanmaktadır.

KİŞİSEL, DAĞITIK ve İSTEMCİ/SUNUCU İŞLEM

1977'de Apple şirketi kişisel bilgisayarları popülerleştirdi. İlk başta hobisel bir yaklaşım olan kişisel bilgisayar kullanımı 1981'de IBM'in Personal Computer'ı tanıtmasıyla sadece evlerde değil iş yaşamında da çok yoğun biçimde kullanılmaya başladı.

Fakat bu bilgisayarlar tek başına birimlerdi. Bilgisayardaki veriyi paylaşabilmek için başka bir bilgisayara taşınabilir disklerle taşımak gerekiyordu. Tek başına, bağımsız üniteler halinde bulunan bilgisayarların LAN veya büyük ağlar aracılığıyla birbirine bağlanmasıyla dağıtık işlem ("distributed processing") başladı. Bugün artık büyük miktarlarda veri, bilgisayar ağlarında yer alan File Server denen ve ortak kullanılacak program ve verileri saklayan bilgisayarlar aracılığıyla istemci bilgisayarlar tarafından kullanılabilir (istemci/sunucu ortam).

C ve C++ günümüzde, bu ortamları (bilgisayar ağları, dağıtık istemci/sunucu ortamlar, bunları destekleyen işletim sistemleri) yazmakta yaygın olarak kullanılan dillerdir.

PROGRAMLAMA DİLLERİ

Programlamanın evrimi bakımında 4 sınıftan bahsedebiliriz:

- 1- Makine dilleri (1.kuşak): sayılardan oluşan, sadece belirli bir makinenin anlayacağı , hata yapmaya çok açık diller
- 2- Assembly dilleri (2.kuşak): İngilizce benzeri komutlardan oluşan diller. Assembly dili ile yazılmış bir program, bir çevirici (translator) ile makine diline çevrilerek çalıştırılır. Oldukça anlaşılabilir gözükmemesine karşın çok basit bir iş için bile çok sayıda komut

yazmak gerekmektedir.

3- Yüksek seviyeli diller (3.kuşak) – “procedural” diller. “NASIL” yapılacağını ayrıntılı olarak bildirdiğimiz diller. İngilizceye benzer, aritmetik işlemlerde kullandığımız işaretleri kullanır, daha hızlı ve kolay program yazılır. C ve C++ bu gruptadır. Çok sayıda dil geliştirilmiştir. Yaygın kabul görenler:

Fortran IBM 1954 Mühendislik uygulamalarında hala yaygın olarak kullanılır.

Cobol 1959 Günümüz ticari uygulamalarının yarısı Cobol ile yazılmıştır.

Pascal Niklaus Wirth 1971 Akademik kullanım için

4- Çok yüksek seviyeli diller (4.kuşak): “NE” yapılacağını söyledğimiz diller: LISP, SNOBOL, FORTH, SQL...

C ve C++'nin TARİHÇESİ

BCPL 1967 Martin Richards

B 1970 Ken Thompson Bell Laboratuvarları

C 1972 Dennis Ritchie Bell ”

UNIX işletim sisteminin yazılmasında kullanıldı. Makineden bağımsız bir dil.

C++ 1980 Bjarne Stroustrup AT&T

1983 – 1989 ANSI C'nin standart tanımını yaptı, 1990'da yayımlandı: ANSI/ISO 9899:1990

C++, C'ye birtakım ek özellikler ile nesne-yönelimli programlama (“object-oriented programming”) yetenekleri getirmiştir. Nesneler tekrar tekrar kullanılabilir yazılım bileşenleridir; yazılım geliştirmede büyük kolaylık (verimlilik) sağlarlar.

Diğer nesne-yönelimli diller:

Smalltalk Xerox's Palo Alto Araştırma Merkezi

Java Sun Microsystems-James Goslin 995 (1993 WWW popülerliği)

C++ STANDART KİTAPLIK

C++ programları sınıflar ("classes") ve fonksiyonlardan ("functions") oluşur. Bunlar program parçalarıdır. Bunların bir kısmını programcı yazar, bir kısmı da standart kitaplıkta hazır bulunur.

YAPISAL PROGRAMLAMA

1960'lara kadar programlama çok karmaşık, zor, maliyetli...

1960'larda yapısal programlama ortaya çıkmaya başladı. **Yapısal programlama**, daha açık, net, anlaşılabilir, okuması, izlemesi, sinaması kolay programlar yazmak için bir yaklaşımdır.

3 temel yapı kullanılır: sıra, tekrar yapısı ve karar verme yapısı. Goto kullanılmaz ya da mümkün olduğunca az kullanılır. Buna uygun ilk dillerden biri (ve en popüler diyebiliriz) Pascal (Niklaus Wirth,1971). Akademik ortamlarda yapısal programlamayı öğretmek için geliştirildi. Pek çok okulda yaygın olarak kullanıldı. Ancak, ticari uygulamalarda ve her türlü iş ortamına uygun uygulamalarda gereksinilen pek çok olanağı içermemesi, üniversite dışında yaygın kabul görmemesine neden olmuştur.

Ada – 1970'ler – 1980'lerin başı – US Dept. Of Defense

ANAHTAR YAZILIM TRENDİ: NESNE TEKNOLOJİSİ

Yazılım teknolojisindeki gelişme ve ilerlemeler; yapısal programlama, yapısal tasarım, yapısal analiz ile başladı. Ancak nesne yönelimli analiz, tasarım ve programlama ile daha çok gelişme elde edildi. Nesne yönelimli ile çevremizdeki tüm nesneleri (araba, öğrenci, ders, insan, uçak, sınıf vb) modelleriz. Modellediğimiz bu yazılım parçalarını tekrar tekrar kullanabildiğimiz gibi, bu tarz yazılan yazılımlar, daha iyi organize ve anlaşılır oldukları için bakımları da çok kolay olmakta ve programcıya zaman ve maliyetten büyük tasarruf sağlamaktadır (Yazılım maliyetinin %80'i programlamanın başında ve yazımındaki emeğe değil sonradan yapılan sürekli bakıma aittir).

C++, nesne yönelimli bir dil olarak anılmasına karşın hibrid bir dildir. Yani hem yapısal hem de nesne yönelimli program yazmayı mümkün kılar

Classes and Data Abstraction konusuna kadar C++ ile "procedural" programlama kavramlarını göreceğiz: kontrol yapıları, fonksiyonlar, veri türleri, giriş/çıkış, diziler, pointer ve stringler. Bu bölümlerde C++'ın C kısmı, C++ procedural enhancements to C kapsanacaktır.

C++ PROGRAMI YÜRÜTME AŞAMALARI

Unix ortamında bir C++ programının yürütülmesi için 6 aşamadan geçilir:

- 1- Edit : kaynak programın bir editör program ile yazılması. Uzantısı .cpp, .cxx veya .c
- 2- Ön işleme ("preprocess"): derleme öncesi bazı işlemler
- 3- Derleme: derlemenin çıktısı: amaç program
- 4- Link: programımızda kullanacağımızı belirttiğimiz standart kitaplık veya özel kitaplıklarda bulunan fonksiyonların amaç programla bir araya getirilmesi (link edilmesi)

5- Yükleme ("load") – programın ve varsa gerekli bazı modüllerin (paylaşılan kitaplıklardan) diskten alınıp belleğe yüklenmesi

6- Yürütme

Şema: s. 15 (Deitel)

TAŞINABİLİRLİK ("portability")

Taşınabilir programlar yazmak mümkünse de değişik sistemler ve değişik derleyiciler arasında birtakım problemler çıkmaktadır. Taşınabilirlikle ilgili konular ANSI C Standart dokümanında yer almaktadır: www.ansi.org Doküman adı: Information Technology – Programming languages – C++. Doküman numarası: ISO/IEC 14882-1998.

Daha eski taslak versiyonu (ücretsiz): www.cygnum.com/misc/wp/

İYİ PROGRAM YAZMA TAVSİYELERİ:

C++ programlarınızı basit ve straightforward biçimde yazın: KIS ("Keep It Simple").

Kullandığınız C++ sürümünün kılavuzu elinizin altında olmalı.

En iyi öğretici bilgisayarınızdır. Kitaplarda veya derslerde işlenen konuları, verilen örnekleri, bilgisayarda çalıştırarak, çıkan hata mesajlarını okuyup, tekrar deneyerek öğrenebilir ve tecrübe kazanabilirsiniz.

BASİT BİR PROGRAM: BİR SATIR METNİN YAZDIRILMASI

```
// C++ ile ilk programım
```

```
#include <iostream>
```

```
int main()
```

```
{
```

```
    std::cout << "Herkes Merhaba!\n";
```

```
    return 0;        // program sonu
```

```
}
```

Herkes Merhaba!

// C++'de açıklama satırının başına konur. Bu satırlar için derleyici bir iş yapmaz; bu satırlar program dökümünde aynen yer alan; programın okunabilirliği, anlaşılabilirliği için gerekli açıklama satırlarıdır. İyi bir programcı gerekli yerlere gerekli açıklamaları koyar.

C dilinde ise açıklama `/*` `*/` arasına konur.

`#` ile başlayan satırlar ön derleyici ("preprocessor") içindir. Örnek programımızda `#` ile başlayan satır ön derleyiciye giriş/çıkış ile ilgili `iostream` başlık dosyasını ("header file") programa dahil etmesini söyler.

int main () her C++ programında olması gerekir.

Her C++ programı bir veya birkaç fonksiyondan oluşur. ()'ler main'in bir fonksiyon olduğunu; "main" sözcüğü de bunun ana fonksiyon olduğunu belirtmektedir.

int integer (tamsayı) anlamında.

int main ()'in anlamı: bu ana fonksiyonun çalıştıktan sonra verdiği değer, bir tamsayıdır.

(return 0 ile dönen sıfır)

Bir fonksiyonun gövdesi ("body") köşeli sol parantez ile { başlar, köşeli sağ parantez ile } biter.

std::cout ekrana çıkış almak için. İlerki bölümlerde daha ayrıntılı göreceğiz.

std::cin klavyeden girileni okumak için

<< "stream insertion operator"

\n karakterlerinin çıktıda yer almadığına dikkat edin!

**** "escape" karakteri

n imlecin alt satıra konumlanmasını söyler.

\n yeni satır

\t bir "tab" ileri gitmek için

\r imleç yeni yazılan sol başına konumlanır ("carriage return")

\a uyarı sesi

Herkese Merhaba! cümlesini başka nasıl yazdırabilirdik?

```
std::cout << "Herkese ";
```

```
std::cout << "Merhaba!\n";
```

ŞÖYLE YAZARSAM ÇIKTI NASIL OLUR???

```
std::cout << "Herkes\nMerhaba!\n";
```

Herkes

e

Merhaba!

BAŞKA BİR BASİT PROGRAM: iki tamsayının toplanması

// İKİ TAMSAYIYI TOPLAYAN PROGRAM

```
#include <iostream>
```

```
int main()
```

```
{
```

```
    int tamsayi1, tamsayi2, toplam; //tanımlamalar
```

```
    std::cout << "Birinci tamsayıyı giriniz\n";
```

```
std::cin >> tamsayi1; //klavyeden girilen sayıyı oku

std::cout << "İkinci tamsayıyı giriniz\n";

std::cin >> tamsayi2; // klavyeden girilen sayıyı oku

toplam = tamsayi1 + tamsayi2;

std::cout << "Toplam = " << toplam << std::endl;


return 0; // programın başarıyla bittiğini gösterir

}
```

std::endl =è endl "end line" yani satır sonu demektir. Yaptığı iş, yeni satırı yazdıktan (ekrana yolladıktan sonra) çıkış buffer'larını temizlemektir. Bilindiği gibi, çıkışa gönderilecek veriler, gönderilmeden önce çıkış buffer'larında biriktirilir. Çıkış işleminden sonra buffer'ların boşaltılması/temizlenmesi için endl kullanırız.

ÖNEMLİ NOTLAR:

C++, büyük ve küçük harfe duyarlı bir dildir.

Cümleler ; (noktalı virgül) ile biter.

İYİ PROGRAM YAZMA TAVSİYELERİ:

- 1- Bir fonksiyonun gövdesinin içine yazılan satırlar 3 boşluk içerden yazılır ("indentation")
- 2- Tanımlamalar ile yürütülebilir cümleler arasında bir satır boşluk bırakınız.
- 3- Anlamlı değişken isimleri kullanınız.
- 4- İkili ("binary") operatörlerin sağ ve solunda birer boşluk bırakınız.

ARİTMETİK OPERATÖRLER

C++ işlemi	Aritmetik operatör	Cebirsel ifade	C++ ifadesi
Toplama	+	$f+7$	$f + 7$
Çıkarma	-	$p-c$	$p - c$
Çarpma	*	bm	$b * m$
Bölme	/	x/y	x / y
Modulus	%	$r \text{ mod } s$	$r \% s$

C++'da aritmetik işlemlerde operatörlerin öncelik sırası kuralları:

1- Parantez içleri en önce hesaplanır.

2- Daha sonra, çarpma, bölme, modulus işlemleri yapılır. Bir aritmetik ifadede birden fazla çarpma, bölme, modulus işlemi varsa, bu işlemlerdeki operatörler SOLDAN SAĞA doğru uygulanır. Çarpma, bölme, mod operatörlerinin önceliği aynıdır.

3- En son toplam ve çıkarma işlemleri yapılır. Bir ifade, birden fazla toplama ve çıkarma işlemi içeriyorsa, bunlara ait operatörler SOLDAN SAĞA uygulanır.

ÖRNEKLER: ss. 31 – 33 (Deitel)

İYİ PROGRAM YAZMA TAVSİYELERİ:

Cebirde olduğu gibi, gereksiz parantezler konmayabilir. Ancak, özellikle uzun ifadelerde alt ifadeleri parantez içlerin alarak anlaşılabilirliği artırmak iyidir.

İLİŞKİSEL OPERATÖRLER ve EŞİTLİK OPERATÖRLERİ

İlişkisel operatörler	C++ karşılığı	C++ örneği	Anlamı
>	>	x > y	x, y'den büyüktür
<	<	x < y	x, y'den küçüktür
eşittir	>=	x >= y	x, y'den büyüktür veya y'ye
eşittir.	<=	x <= y	x, y'den küçüktür veya y'ye

Eşitlik operatörleri

=	==	x ==y	x, y'ye eşittir
	!=	x != y	x, y'ye eşit değildir

SIK YAPILAN HATALAR:

==, >=, <=, != operatörlerini yazarken iki karakter arasında boşluk bırakılmamalıdır veya iki karakterin yeri değiştirilmemelidir.

Eşitlik operatörü '==' yi, atama için kullanılan '=' ile karıştırmayınız.

EŞİTLİK VE BAĞINTI OPERATÖRLERİNİ KULLANAN BİR PROGRAM ÖRNEĞİ

// if cümlesi ile eşitlik ve ilişki/bağıntı operatörlerini kullanan bir program

```
#include <iostream>
```

```
using std::cout; // program cout kullanır
```

```
using std::cin; // program cin kullanır
```

```
using std::endl; // program endl kullanır.
```

```
int main ()
```

```
{
```

```
    int sayi1, sayi2;
```

```
    cout << "İki tamsayı giriniz.\n"
```

```
<< "Ben aralarındaki ilişkiyi söyleyeceğim: ";
```

```
    cin >> sayi1 >> sayi2;
```

```
    if (sayi1 == sayi2)
```

```
        cout << sayi1 << sayi2 << "'ye eşittir." << endl;
```

```
    if (sayi1 != sayi2)
```

```
        cout << sayi1 << sayi2 << "'ye eşit değildir." << endl;
```

```
    if (sayi1 < sayi2)
```

```
    cout << sayi1 << ", " << sayi2 << "'den küçüktür." << endl;

    if (sayi1 > sayi2)

        cout << sayi1 << ", " << sayi2 << "'den büyüktür." << endl;

    if (sayi1 <= sayi2)

        cout << sayi1 << ", " << sayi2 << "'den küçük veya eşittir." << endl;

    if (sayi1 >= sayi2)

        cout << sayi1 << ", " << sayi2 << "'den büyük veya eşittir." << endl;

    return 0;

}
```

İki tamsayı giriniz.

Ben size aralarındaki ilişkiyi söyleyeceğim: 3 7

3, 7'ye eşit değildir.

3, 7'den küçüktür.

3, 7'den küçük veya eşittir.

İki tamsayı giriniz.

Ben size aralarındaki ilişkiyi söyleyeceğim: 22 12

22, 12'ye eşit değildir.

22, 12'den büyüktür.

22, 12'den büyük veya eşittir.

İki tamsayı giriniz.

Ben size aralarındaki ilişkiyi söyleyeceğim: 7 7

7, 7'ye eşittir.

7, 7'den küçük veya eşittir.

7, 7'den büyük veya eşittir.

Using cümleleri: `cout` ve `cin`'den önce `std::` kullanımını ortadan kaldırır. Hatırlanacağı gibi ilk basit örneğimizde `std::cout` kullanmıştık. Çok sayıda giriş / çıkış yapılacağı zaman her `cout` ve `cin`'den önce `std::` kullanmak çok pratik olmayacaktır. Bu nedenle `using` kullanmakta fayda vardır.

İYİ PROGRAM YAZMA TAVSİYELERİ:

- 1- Programın okunabilirliğini artırmak için `if` yapısının içinde yer alan cümleleri 3 boşluk içerden yazınız.
- 2- Bir satırda birden fazla cümle yer almamalı.
- 3- Bir cümle bir satıra sığmayıp alt satırdan devam edecekse, cümleyi anlamlı bir yerinden bölmeye özen gösteriniz.

Etiketler:[c++](#), [c++ dersleri](#), [c++ dosyaları](#), [c++ nedir](#), [c++ programlama](#), [c++ tarihi](#), [C++ İLE PROGRAMLAMAYA GİRİŞ](#), [programalama dilleri](#)